**CHAPTER 1**

**INTRODUCTION**

* 1. **INTRODUCTION TO SQL**

Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's Relational model of database. Today almost all RDBMS(MySql, Oracle, Infomix, Sybase, MS Access) use SQL as the standard database query language. SQL is used to perform all types of data operations in RDBMS. SQL allows users to query the database in a number of ways, using English-like statements.

SQL statement tells the database that what information you would like to retrieve or what operation you want to perform on the data. A database is a organized collection of data. Data is stored in relational database in form of tables. To create, retrieve, update and delete from relational database and tables.

SQL defines following data languages to manipulate data of RDBMS:

1. DDL: Data Definition Language

All DDL commands are auto-committed. That means it saves all the changes permanently in the database.

Eg: create-To create new table or database, alter-For alteration, truncate-Delete from table, drop-To drop a table

2. DML: Data Manipulation Language

DML commands are not auto-committed. It means changes are not permanent to database, they can be rolled back.

Eg : insert-To insert a new row, update-To update existing row, delete-To delete a row, merge-merging two rows or two tables

3. TCL: Transaction Control Language

These commands are to keep a check on other commands and their affect on the database. These commands can annul changes made by other commands by rolling back to original state. It can also make changes permanent.

Eg : commit-to permanently save, rollback-to undo change, save point-to save temporarily

4. DCL: Data Control Language

Data control language provides command to grant and take back authority.

Eg : grant-grant permission of right, revoke-take back permission

5. DQL-Data Query Language

DQL is used to operate on queries.

Eg : Select-retrieve records from one or more table

**1.2 INTRODUCTION TO FRONT END SOFTWARE**

The front end software used here is Python. Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

**1.3 PROJECT REPORT OUTLINE**

The report is arranged in the following way:

Chapter 1: Gives the information of the project

Chapter 2: Gives the Requirement Specification of the project

Chapter 3: Gives the Objective of the project

Chapter 4: Gives the Implementation of the project

Chapter 5: Gives the Front end design of the project

Chapter 6: Gives the Testing information of the project

Chapter 7: Gives the Results of the project

**CHAPTER 2**

**REQUIREMENT SPECIFICATION**

**2.1 SOFTWARE REQUIREMENTS**

Operating System : Windows 10

Database : SQLite

Tools : Python

**2.2 HARDWARE REQUIREMENTS**

Processor : Intel i5 8th Generation

RAM : 8 GB

Hard Disk : 1 TB

Compact Disk : CD-ROM, CD-R, CD-RW

Input device : Keyboard, Mouse

Output device : Monitor

**CHAPTER 3**

**OBJECTIVE OF THE PROJECT**

The main objective of the Criminal Database Management is to manage the Criminal information like Criminal’s name, Criminal’s Age, Crime, Date of crime, etc. It even manages the Victims Data such as Name, Age, Address. This Application is build for the Department and hence, only Police Officers and Department People can access the Private information.

The purpose of the project is to reduce manly work and make it easy to store and access the information for the Department.

The system includes different functional divisions which are-

* Insertion of Criminal’s information
* Accessing and Viewing the information
* Deletion of information
* Searching from the Database
* Update the information
* Accessing the Victim’s Information

The project shows information and description of criminals and victims thus increasing the efficiency of managing. It deals with monitoring the information for the Department. Adding, Deleting, Searching and Updating of records is improved which results in proper resource management for the Department.

**CHAPTER 4**

**IMPLEMENTATION**

**4.1 ER DIAGRAM**

**Diagram

Description automatically generated**

**4.2 MAPPING OF ER DIAGRAM TO SCHEMA**

**DIAGRAM**

**STEP 1: MAPPING OF REGULAR ENTITIES**

For each regular (Strong) entity type E in the ER schema.

Create a relation R that include all simple attributes of E. Regular entities of this Criminal database are Accounts, Branch, CriminalInfo and Victims.

**Accounts**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| policeID | name | age | email | password |

**Branch**

|  |  |
| --- | --- |
| branchID | branch\_name |

**CriminalInfo**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| criminalID | name | age | crime | date | victim |

**Victim**

|  |  |  |
| --- | --- | --- |
| victimID | victim | victim\_age |

**STEP 2: MAPPING OF WEAK ENTITIES**

A weak entity cannot be used independently as it is dependent on a strong entity type known as its owner entity. Also, the relationship that connects the weak entity to its owner identity is called the identifying relationship.

A weak entity always has a total participation constraint with respect to its identifying relationship because it cannot be identified independently of its owner identity.

A weak entity may have a partial key, which is a list of attributes that identify weak entities related to the same owner entity.

Weak entity in this Criminal Database is Address

**Address**

|  |  |
| --- | --- |
| victimID | victim\_address |

**STEP 3: MAPPING OF 1:1 RELATION TYPE**

For each binary 1:1 relationship type are in the ER schema identify the relation S and T

That correspond to the entity type participating in are. There are three are possible approaches

The foreign key approach, The merged relationship approach, The cross reference or relationship relation approach

Foreign key approach: Chooses one of the relation S and include as a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S include all the simple attribute of the 1:1 relationship type R as a attribute of S.

**Victim**

|  |  |  |
| --- | --- | --- |
| victimID | victim | victim\_age |

**Address**

|  |  |
| --- | --- |
| victimID | victim\_address |

**STEP 4: MAPPING OF 1:N RELATION TYPE**

For each regular binary 1: N relationship type R.

a) Identify the relation S that represents the participating entity type at the N-side of the relationship type.

b)Include as foreign key in S the primary key of the relations T that represents the other entity type participating in R.

c) Include any simple attributes of the 1:N relationship type as attribute of S.

**Branch**

|  |  |
| --- | --- |
| branchID | branch\_name |

**CriminalInfo**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| criminalID | name | age | crime | date | victim |

**STEP 5: MAPPING OF M:N RELATION TYPE**

For each binary M:N type R

A) Create new relation S to represent R

B) Include a foreign key attributes in S the primary keys of the relations that represent the participating entity types their combinations will form the primary key of S

C) Also, include any simple attributes of the M:N relationship type as attributes of S

**Accounts**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| policeID | name | age | email | password |

**Branch**

|  |  |
| --- | --- |
| branchID | branch\_name |

**CriminalInfo**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| criminalID | name | age | crime | date | victim |

**Victim**

|  |  |  |
| --- | --- | --- |
| victimID | victim | victim\_age |

**4.3 MAPPING OF THE ER SCHEMA TO RELATIONS**

**Accounts**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| policeID | name | age | email | password |

**Branch**

|  |  |
| --- | --- |
| branchID | branch\_name |

**CriminalInfo**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| criminalID | name | age | crime | date | victim |

**Victim**

|  |  |  |
| --- | --- | --- |
| victimID | victim | victim\_age |

**Address**

|  |  |
| --- | --- |
| victimID | victim\_address |

**4.4 NORMALIZE**

Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update and anomaly.

**FIRST NORMAL FORM (1NF)**

A Data is said to be in first normal form if:

1. There are no duplicate rows in the table.

2. Each cell is single valued or atomic

**SECOND NORMAL FORM (2NF)**

A Database is said to be in second normal form if

1.If the tables are in 1NF and if all non-key attributes of the tables are fully functionally dependent on all the key attributes.

**THIRD NORMAL FORM (3NF)**

A Database is said to be in second normal form if

1. If all the tables in it are in 2NF and without any transitive dependencies, i.e X>Y, Y>Z, X>Z.

2. According to CODD‟s definition a relation schema R is in 3NF if it satisfies 2NF and all non-prime attributes are transitively dependent on the primary key.

**4.5 CREATION OF TABLES**

**CREATION OF ACCOUNTS TABLE**

CREATE TABLE Accounts (

policeID INTEGER PRIMARY KEY AUTOINCREMENT,

name text NOT NULL,

age Number(2) NOT NULL,

email text NOT NULL,

passw text NOT NULL);

**CREATION OF BRANCH TABLE**

CREATE TABLE Branch (

branchID INTEGER PRIMARY KEY AUTOINCREMENT,

branch\_name Varchar NOT NULL);

**CREATION OF CRIMINAL INFORMATION TABLE**

CREATE TABLE CriminalInfo (

criminalID INTEGER PRIMARY KEY AUTOINCREMENT,

name text NOT NULL,

age Number(2) NOT NULL,

crime text NOT NULL,

date Date NOT NULL,

victim text NOT NULL);

**CREATION OF VICTIMS TABLE**

CREATE TABLE Victims (

victimID INTEGER PRIMARY KEY AUTOINCREMENT,

victim text NOT NULL,

victim\_age Number(2) NOT NULL);

**CREATION OF ADDRESSES TABLE**

CREATE TABLE Address (

victimID References Victims (victimID) On Delete Cascade,

victim\_address text NOT NULL);

**4.6 INSERTION OF TUPLES**

**INSERTING INTO BRANCH TABLE**

INSERT INTO Branch (branch\_name) VALUES("Koramangala");

INSERT INTO Branch (branch\_name) VALUES("Indiranagar");

INSERT INTO Branch (branch\_name) VALUES("Rajajinagar");

INSERT INTO Branch (branch\_name) VALUES("MG Road");

INSERT INTO Branch (branch\_name) VALUES("HSR Layout");

INSERT INTO Branch (branch\_name) VALUES("RR Nagar");

INSERT INTO Branch (branch\_name) VALUES("Whitefield");

INSERT INTO Branch (branch\_name) VALUES("BTM Layout");

INSERT INTO Branch (branch\_name) VALUES("Jayanagar");

INSERT INTO Branch (branch\_name) VALUES("Electronic City");

INSERT INTO Branch (branch\_name) VALUES("Bellandur");

INSERT INTO Branch (branch\_name) VALUES("Banashankari");

**INSERTING INTO CRIMINALINFO TABLE**

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Manish", 56, "Theft", "12/31/20", "Shreyas");

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Kamal", 43, "Murder", "12/4/19", "Sonal");

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Sonam", 32, "Robbery", "10/16/19", "Komal");

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Rajesh", 34, "Child Abuse", "12/31/20", "Sunali");

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Abhimanyu", 29, "Kidnapping", "12/16/19", "Kartik");

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Keshav", 39, "Genocide", "12/31/20", "Aditya and Others");

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Aadi", 51, "Torture", "7/17/19", "Bhuvan");

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Gautami", 21, "Theft", "5/8/19", "Abhyuday");

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Aaquib", 43, "Human Trafficking", "11/6/19", "Sana and Others");

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Amit", 29, "Rape", "11/22/18", "Komal");

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Ajey", 35, "Arson", "1/17/18", "Abhishek");

INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES("Ashish", 45, "Drug Trafficking", "3/21/18", "So Many");

**INSERTING INTO VICTIMS TABLE**

INSERT INTO Victims (victim, victim\_age) VALUES("Abhyuday", 25);

INSERT INTO Victims (victim, victim\_age) VALUES("Bhuvan", 34);

INSERT INTO Victims (victim, victim\_age) VALUES("Komal", 27);

INSERT INTO Victims (victim, victim\_age) VALUES("Kartik", 43);

INSERT INTO Victims (victim, victim\_age) VALUES("Avneet", 35);

INSERT INTO Victims (victim, victim\_age) VALUES("Abhimanyu", 42);

INSERT INTO Victims (victim, victim\_age) VALUES("Manish", 37);

INSERT INTO Victims (victim, victim\_age) VALUES("Ashish", 23);

INSERT INTO Victims (victim, victim\_age) VALUES("Abhishek", 45);

INSERT INTO Victims (victim, victim\_age) VALUES("Sonal", 15);

INSERT INTO Victims (victim, victim\_age) VALUES("Siddhant", 38);

INSERT INTO Victims (victim, victim\_age) VALUES("Sunali", 32);

INSERT INTO Victims (victim, victim\_age) VALUES("Shreyas", 26);

INSERT INTO Victims (victim, victim\_age) VALUES("Keshav", 58);

INSERT INTO Victims (victim, victim\_age) VALUES("Rajesh", 35);

INSERT INTO Victims (victim, victim\_age) VALUES("Faizal", 44);

**INSERTING INTO ADDRESS TABLE**

INSERT INTO Address VALUES (1, "304 MG Road");

INSERT INTO Address VALUES (2, "67 Koramangala");

INSERT INTO Address VALUES (3, "5 Feet Road");

INSERT INTO Address VALUES (4, "RR Nagar");

INSERT INTO Address VALUES (5, "97 Jayanagar");

INSERT INTO Address VALUES (6, "Electronic City");

INSERT INTO Address VALUES (7, "21 Koramangala");

INSERT INTO Address VALUES (8, "RR Nagar");

INSERT INTO Address VALUES (9, "Gulbarga");

INSERT INTO Address VALUES (10, "86 MG Road");

INSERT INTO Address VALUES (11, "Kolar");

INSERT INTO Address VALUES (12, "HSR Layout");

INSERT INTO Address VALUES (13, "25 Banashankari");

INSERT INTO Address VALUES (14, "Hyderabad");

INSERT INTO Address VALUES (15, "92 Rajajinagar");

INSERT INTO Address VALUES (16, "Tumkur");

**4.7 CREATION OF TRIGGERS**

A trigger is a special kind of a store procedure that executes in response to certain action on the table like insertion, deletion or updation of data.

**CREATING A TRIGGER WHICH RAISES AN ERROR WHEN ENTERED INCORRECT EMAIL FORMAT**

CREATE TRIGGER trigg1

BEFORE INSERT ON Accounts

BEGIN SELECT CASE WHEN NEW.email NOT LIKE "%\_@\_%.\_%"

THEN RAISE (ABORT, "Invalid")

END;

END;

**CREATING A TRIGGER WHICH RAISES AN ERROR WHEN ENTERED AN AGE BELOW 18 AND ABOVE 70**

CREATE TRIGGER trigg2

BEFORE INSERT ON Accounts

BEGIN SELECT CASE WHEN NEW.age NOT BETWEEN 18 AND 70

THEN RAISE (ABORT, "Invalid")

END;

END;

**CHAPTER 5**

**FRONT END DESIGN**

**5.1 CONNECTIVITY TO DATABASE**

## **Create Connection**

To use SQLite3 in Python, first of all, you will have to import the sqlite3 module and then create a connection object which will connect us to the database and will let us execute the SQL statements.

You can a connection object using the connect() function:

import sqlite3

conn = sqlite3.connect(‘myfile.db’)

## **SQLite3 Cursor**

To execute SQLite statements in Python, you need a cursor object. You can create it using the cursor() method.

The SQLite3 cursor is a method of the connection object. To execute the SQLite3 statements, you should establish a connection at first and then create an object of the cursor using the connection object as follows:

conn = sqlite3.connect(‘myfile.db’)

curs = conn.curs()

## **Close Connection**

Once you are done with your database, it is a good practice to close the connection. You can close the connection by using the close() method.

To close a connection, use the connection object and call the close() method as follows:

conn.close()

**5.2 FRONT END CODE**

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

Until the 1990s systems design had a crucial and respected role in the data processing industry. In the 1990s standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering.

Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design.[citation needed] The UML has become the standard language in object-oriented analysis and design.[citation needed] It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.[citation needed]

System design is one of the most important phases of software development process. The purpose of the design is to plan the solution of a problem specified by the requirement documentation. In other words the first step in the solution to the problem is the design of the project.

**FRONT END CODE FOR INSERTING THE DATA IN DATABASE:**

def insert(n):

f4 = Frame(bg='#aaaaff')

n.add(f4, text="Insert")

s1 = StringVar()

s2 = StringVar()

s3 = StringVar()

s4 = StringVar()

s5 = StringVar()

u0 = Label(f4, text='Enter Criminal Information', bg='#aaaaff')

u0.config(font=('eras medium itc', 25))

u0.place(x=300, y=100)

u2 = Label(f4, font=('eras medium itc', 20), bg='#aaaaff', text="Enter Name")

u2.place(x=300, y=200)

e2 = Entry(f4, font=('eras medium itc', 12), textvariable=s1)

e2.place(x=500, y=210)

u3 = Label(f4, font=('eras medium itc', 20), bg='#aaaaff', text="Enter Age")

u3.place(x=300, y=250)

e3 = Entry(f4, font=('eras medium itc', 12), textvariable=s2)

e3.place(x=500, y=250)

u5 = Label(f4, font=('eras medium itc', 20), bg='#aaaaff', text="Enter Date")

u5.place(x=300, y=300)

e5 = DateEntry(f4, font=('eras medium itc', 12), textvariable=s4)

e5.place(x=500, y=310)

u4 = Label(f4, font=('eras medium itc', 20), bg='#aaaaff', text="Crime")

u4.place(x=300, y=350)

e4 = Entry(f4, font=('eras medium itc', 12), textvariable=s3)

e4.place(x=500, y=360)

u6 = Label(f4, font=('eras medium itc', 20), bg='#aaaaff', text="Victim Name")

u6.place(x=300, y=400)

e6 = Entry(f4, font=('eras medium itc', 12), textvariable=s5)

e6.place(x=500, y=410)

def insert1():

db = sqlite3.connect('myfile.db')

cr = db.cursor()

cr.execute("INSERT INTO CriminalInfo (name, age, crime, date, victim) VALUES ('" + s1.get() + "','" + s2.get() + "','" + s3.get() + "','" + s4.get() + "','" + s5.get() + "')")

db.commit()

db.close()

messagebox.showinfo("Action completed", "Insertion Executed")

s1.set("")

s2.set("")

s3.set("")

s4.set("")

s5.set("")

show1(f55)

b1 = Button(f4, text="Insert", command=insert1, height=2, width=15)

b1.place(x=450, y=500)

**FRONT END CODE FOR VIEWING THE DATA FROM DATABASE:**

def show(n):

f5 = Frame(bg='#aaaaff')

n.add(f5, text="Show All")

global f55

f55 = f5

show1(f5)

def show1(f5):

for w in f5.winfo\_children():

w.destroy()

u1 = Label(f5, text="ID", font=('eras medium itc', 12), bg="#000000", fg="white")

u1.place(x=0, y=10, width=100)

u2 = Label(f5, text="Name", font=('eras medium itc', 12), bg="#000000", fg="white")

u2.place(x=100, y=10, width=220)

u3 = Label(f5, text="Age", font=('eras medium itc', 12), bg="#000000", fg="white")

u3.place(x=320, y=10, width=100)

u4 = Label(f5, text="Crime", font=('eras medium itc', 12), bg="#000000", fg="white")

u4.place(x=420, y=10, width=220)

u5 = Label(f5, text="Date", font=('eras medium itc', 12), bg="#000000", fg="white")

u5.place(x=640, y=10, width=150)

u6 = Label(f5, text="Victim", font=('eras medium itc', 12), bg="#000000", fg="white")

u6.place(x=790, y=10, width=210)

db = sqlite3.connect('myfile.db')

cr = db.cursor()

r = cr.execute("SELECT \* FROM CriminalInfo")

x = 0

y = 50

for r1 in r:

Label(f5, text=r1[0], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=100)

x = x + 100

Label(f5, text=r1[1], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=220)

x = x + 220

Label(f5, text=r1[2], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=100)

x = x + 100

Label(f5, text=r1[3], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=220)

x = x + 220

Label(f5, text=r1[4], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=150)

x = x + 150

Label(f5, text=r1[5], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=210)

y = y + 30

x = 0

db.commit()

db.close()

**FRONT END CODE FOR SEARCHING THE DATA FROM DATABASE:**

def search(n):

f6 = Frame(bg='#aaaaff')

n.add(f6, text="Search")

s1 = StringVar()

u0 = Label(f6, font=('eras medium itc', 20), bg='#aaaaff', text="Enter Criminal Name")

u0.place(x=200, y=50)

e0 = Entry(f6, font=('eras medium itc', 20), textvariable=s1)

e0.place(x=500, y=50)

def search1():

u1 = Label(f6, text="ID", font=('eras medium itc', 12), bg="#000000", fg="white")

u1.place(x=0, y=200, width=100)

u2 = Label(f6, text="Name", font=('eras medium itc', 12), bg="#000000", fg="white")

u2.place(x=100, y=200, width=220)

u3 = Label(f6, text="Age", font=('eras medium itc', 12), bg="#000000", fg="white")

u3.place(x=320, y=200, width=100)

u4 = Label(f6, text="Crime", font=('eras medium itc', 12), bg="#000000", fg="white")

u4.place(x=420, y=200, width=220)

u5 = Label(f6, text="Date", font=('eras medium itc', 12), bg="#000000", fg="white")

u5.place(x=640, y=200, width=150)

u6 = Label(f6, text="Victim", font=('eras medium itc', 12), bg="#000000", fg="white")

u6.place(x=790, y=200, width=210)

db = sqlite3.connect('myfile.db')

cr = db.cursor()

r = cr.execute("SELECT \* FROM CriminalInfo WHERE name='" + s1.get() + "'")

x = 0

y = 250

for r1 in r:

Label(f6, text=r1[0], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=100)

x = x + 100

Label(f6, text=r1[1], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=220)

x = x + 220

Label(f6, text=r1[2], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=100)

x = x + 100

Label(f6, text=r1[3], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=220)

x = x + 220

Label(f6, text=r1[4], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=150)

x = x + 150

Label(f6, text=r1[5], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=210)

break

else:

messagebox.showerror('Action Failed', 'Invalid Name')

db.commit()

db.close()

b1 = Button(f6, text='Search', command=search1, height=2, width=15)

b1.place(x=400, y=100)

**FRONT END CODE FOR UPDATING THE DATA IN DATABASE:**

def update(n):

f7 = Frame(bg='#aaaaff')

n.add(f7, text="Update")

s1 = StringVar()

u0 = Label(f7, font=('eras medium itc', 20), bg='#aaaaff', text="Enter Criminal ID")

u0.place(x=200, y=50)

e0 = Entry(f7, font=('eras medium itc', 20), textvariable=s1)

e0.place(x=500, y=50)

def update1():

db = sqlite3.connect('myfile.db')

cr = db.cursor()

r = cr.execute("SELECT \* FROM CriminalInfo WHERE criminalID='" + s1.get() + "'")

for r1 in r:

s2 = StringVar()

s3 = StringVar()

s4 = StringVar()

s5 = StringVar()

s6 = StringVar()

u2 = Label(f7, font=('eras medium itc', 20), bg='#aaaaff', text="Enter Name")

u2.place(x=300, y=250)

e2 = Entry(f7, font=('eras medium itc', 12), textvariable=s2)

e2.place(x=500, y=260)

e2.insert(0, r1[1])

u3 = Label(f7, font=('eras medium itc', 20), bg='#aaaaff', text="Enter Age")

u3.place(x=300, y=300)

e3 = Entry(f7, font=('eras medium itc', 12), textvariable=s3)

e3.place(x=500, y=310)

e3.insert(0, r1[2])

u5 = Label(f7, font=('eras medium itc', 20), bg='#aaaaff', text="Enter Date")

u5.place(x=300, y=350)

e5 = DateEntry(f7, font=('eras medium itc', 12), textvariable=s5)

e5.place(x=500, y=360)

u4 = Label(f7, font=('eras medium itc', 20), bg='#aaaaff', text="Crime")

u4.place(x=300, y=400)

e4 = Entry(f7, font=('eras medium itc', 12), textvariable=s4)

e4.place(x=500, y=410)

e4.insert(0, r1[3])

u6 = Label(f7, font=('eras medium itc', 20), bg='#aaaaff', text="Victim Name")

u6.place(x=300, y=450)

e6 = Entry(f7, font=('eras medium itc', 12), textvariable=s6)

e6.place(x=500, y=460)

e6.insert(0, r1[5])

def update2():

db = sqlite3.connect('myfile.db')

cr = db.cursor()

cr.execute("UPDATE CriminalInfo SET name ='" + s2.get() + "', age ='" + s3.get() + "', crime ='" + s4.get() + "', date ='" + s5.get() + "', victim ='" + s6.get() + "' where CriminalID ='" + s1.get() + "'")

db.commit()

db.close()

show1(f55)

messagebox.showinfo('Action Completed', 'Criminal Information Updated')

b11 = Button(f7, text="Update", command=update2, height=2, width=15)

b11.place(x=450, y=550)

break

else:

messagebox.showerror('Action Failed', 'Invalid Criminal ID')

db.commit()

db.close()

b1 = Button(f7, text='Fetch', command=update1, height=2, width=15)

b1.place(x=400, y=100)

**FRONT END CODE FOR DELETING THE DATA FROM DATABASE:**

def delete(n):

f8 = Frame(bg='#aaaaff')

n.add(f8, text="Delete")

s1 = StringVar()

u0 = Label(f8, font=('eras medium itc', 20), bg='#aaaaff', text="Enter Criminal ID")

u0.place(x=200, y=50)

e0 = Entry(f8, font=('eras medium itc', 20), textvariable=s1)

e0.place(x=500, y=50)

def delete1():

u1 = Label(f8, text="ID", font=('eras medium itc', 12), bg="#000000", fg="white")

u1.place(x=0, y=200, width=100)

u2 = Label(f8, text="Name", font=('eras medium itc', 12), bg="#000000", fg="white")

u2.place(x=100, y=200, width=220)

u3 = Label(f8, text="Age", font=('eras medium itc', 12), bg="#000000", fg="white")

u3.place(x=320, y=200, width=100)

u4 = Label(f8, text="Crime", font=('eras medium itc', 12), bg="#000000", fg="white")

u4.place(x=420, y=200, width=220)

u5 = Label(f8, text="Date", font=('eras medium itc', 12), bg="#000000", fg="white")

u5.place(x=640, y=200, width=150)

u6 = Label(f8, text="Victim", font=('eras medium itc', 12), bg="#000000", fg="white")

u6.place(x=790, y=200, width=210)

db = sqlite3.connect('myfile.db')

cr = db.cursor()

r = cr.execute("SELECT \* FROM CriminalInfo WHERE criminalID='" + s1.get() + "'")

x = 0

y = 250

for r1 in r:

Label(f8, text=r1[0], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=100)

x = x + 100

Label(f8, text=r1[1], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=220)

x = x + 220

Label(f8, text=r1[2], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=100)

x = x + 100

Label(f8, text=r1[3], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=220)

x = x + 220

Label(f8, text=r1[4], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=150)

x = x + 150

Label(f8, text=r1[5], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=210)

break

else:

messagebox.showerror('Action Failed', 'Invalid Criminal ID')

db.commit()

db.close()

def delete2():

db = sqlite3.connect('myfile.db')

cr = db.cursor()

cr.execute(" DELETE FROM CriminalInfo WHERE criminalID='" + s1.get() + "'")

db.commit()

db.close()

show1(f55)

messagebox.showinfo('Action Completed', 'Data Deleted')

s1.set("")

btn = Button(f8, text='Confirm', command=delete2, height=2, width=15)

btn.place(x=400, y=500)

b1 = Button(f8, text='Fetch', command=delete1, height=2, width=15)

b1.place(x=400, y=100)

**FRONT END CODE FOR VIEWING THE VICTIM’S DATA FROM DATABASE:**

def victim(n):

f9 = Frame(bg='#aaaaff')

n.add(f9, text="Victims")

u0 = Label(f9, text="List of all Victims", font=('eras medium itc', 20), bg='#aaaaff')

u0.place(x=400, y=30)

u1 = Label(f9, text="ID", font=('eras medium itc', 12), bg="#000000", fg="white")

u1.place(x=0, y=100, width=100)

u2 = Label(f9, text="Name", font=('eras medium itc', 12), bg="#000000", fg="white")

u2.place(x=100, y=100, width=300)

u3 = Label(f9, text="Age", font=('eras medium itc', 12), bg="#000000", fg="white")

u3.place(x=400, y=100, width=200)

u4 = Label(f9, text="Address", font=('eras medium itc', 12), bg="#000000", fg="white")

u4.place(x=600, y=100, width=400)

db = sqlite3.connect('myfile.db')

cr = db.cursor()

r = cr.execute("SELECT \* FROM victims")

x = 0

y = 150

for r1 in r:

Label(f9, text=r1[0], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=100)

x = x + 100

Label(f9, text=r1[1], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=300)

x = x + 300

Label(f9, text=r1[2], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=200)

y = y + 30

x = 0

r2 = cr.execute("SELECT victim\_address from Address")

x = 600

y = 150

for r3 in r2:

Label(f9, text=r3[0], font=('eras medium itc', 12), bg="#000000", fg="white").place(x=x, y=y, width=400)

y = y + 30

db.commit()

db.close()

**CHAPTER 6**

**TESTING**

This chapter gives the outline of all testing methods that are carried out to get a bug free system. Quality can be achieved by testing the product using different techniques at different phases of the project development. The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components sub assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

* 1. **TESTING PROCESS**

Testing is an integral part of software development. Testing process certifies whether the product that is developed compiles with the standards that it was designed to. Testing process involves building of test cases against which the product has to be tested.

* 1. **TESTING OBJECTIVES**

The main objectives of testing process are as follows.

* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has high probability of finding undiscovered error.
* A successful test is one that uncovers the undiscovered error.

**6.3 TEST CASES**

The test cases provided here test the most important features of the project.

**6.3.1 Test cases for the project**

**Table 6.1 ------- Test Case**

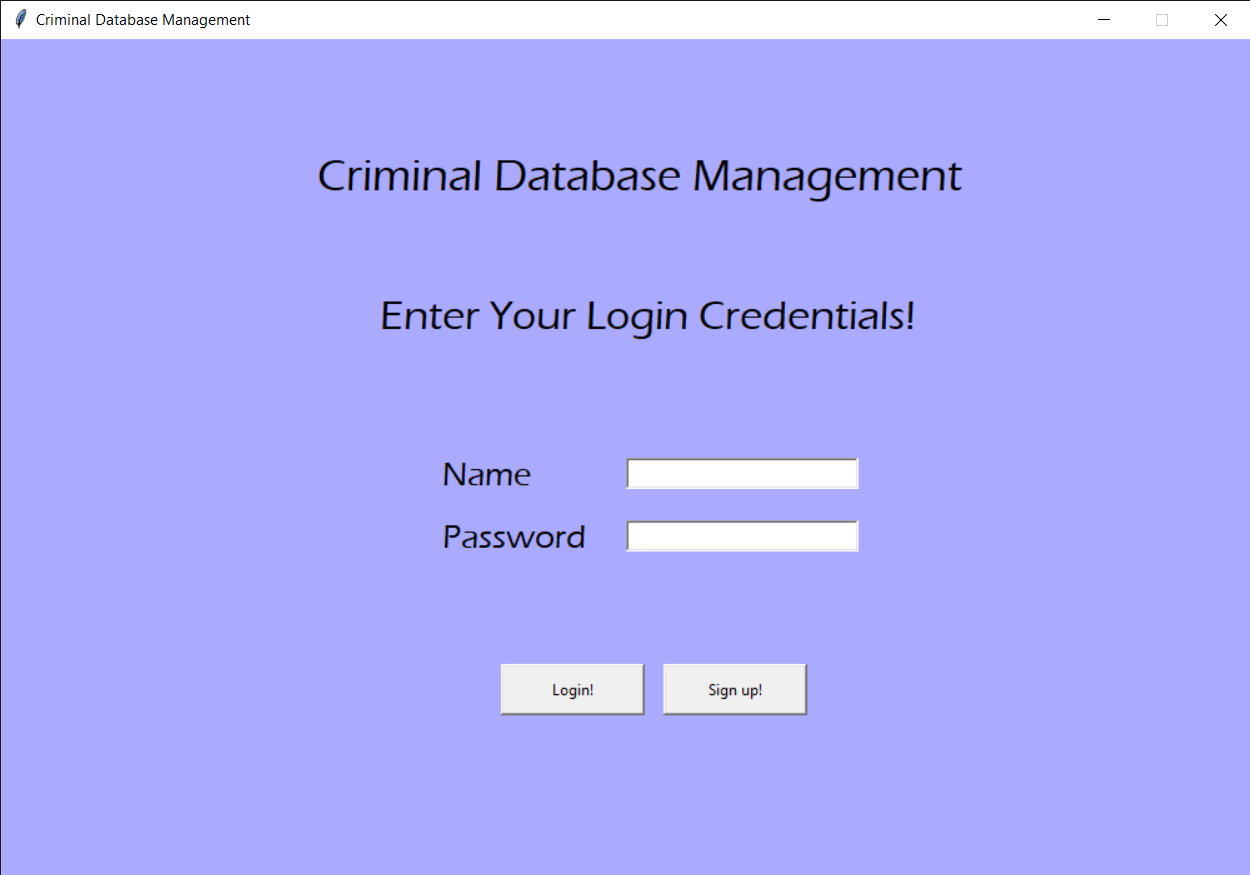
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No** | **Test Input** | **Expected Results** | **Observed Results** | **Remarks** |
| 1 | Insert a record | New tuple should be inserted. | New record inserted | PASS |
| 2 | Search a record | Should display the searched record | Searched record found | PASS |
| 3 | Search a record | Should display the searched record | Searched record not found | FAIL |
| 4 | Update a record | Update the record | Record Updated | PASS |
| 5 | Logout | Move to the home page | Logged out | PASS |
| 6 | Delete a record | Delete the record | Record not found | FAIL |
| 7 | Delete a record | Delete the record | Selected record deleted | PASS |

**CHAPTER 7**

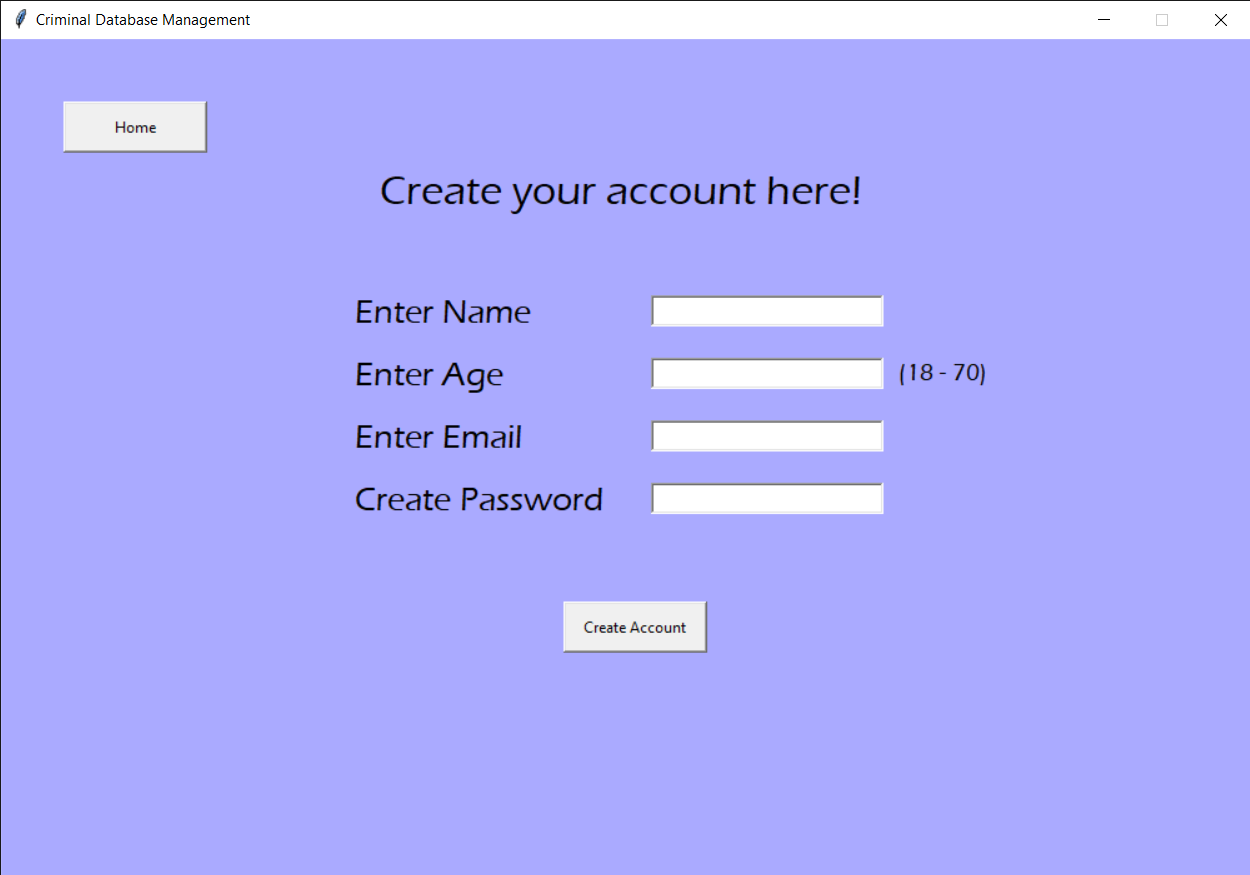
**RESULTS**

This section describes the screens of the “Criminal Database Management”. The snapshots are shown below for each module.

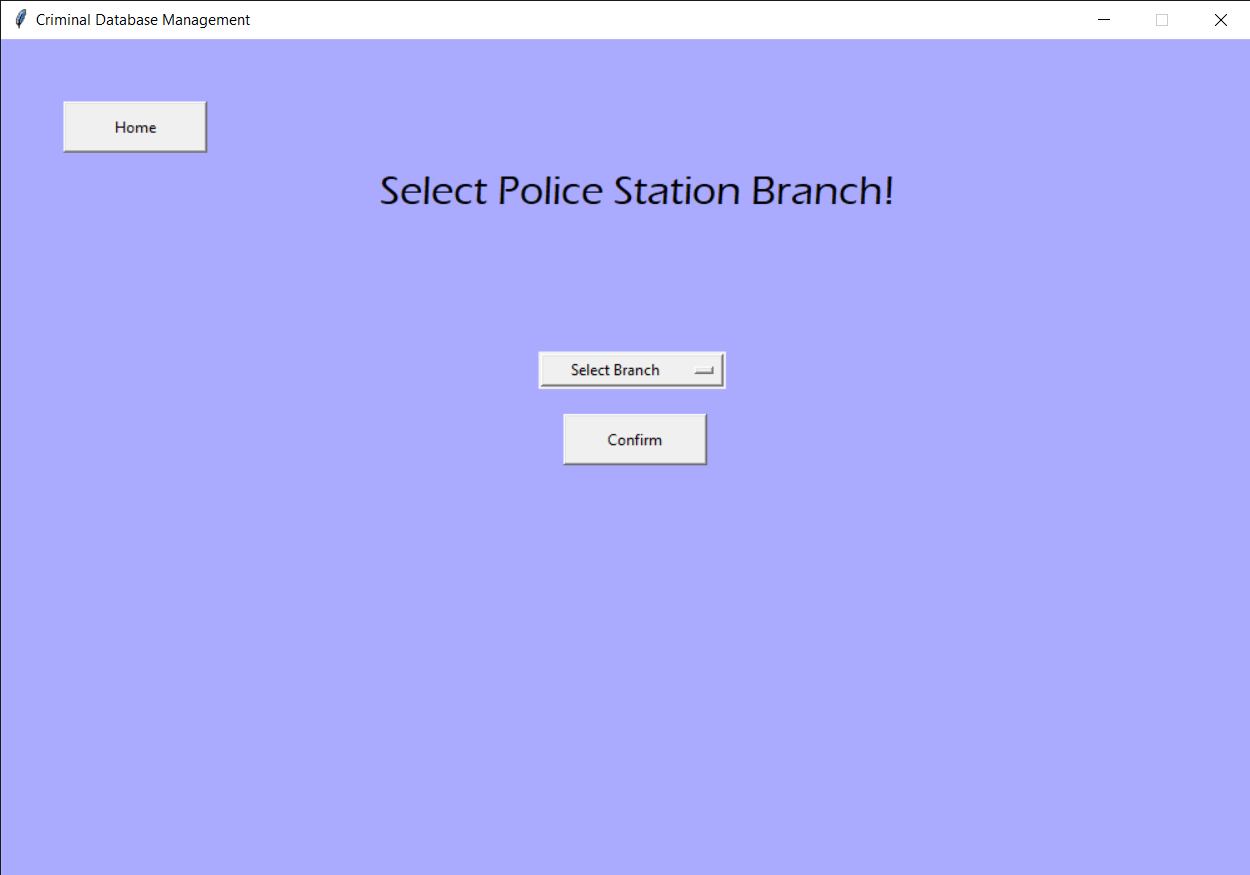
**7.1 SNAPSHOTS**

****

**Snapshot 1: Login Page**

****

**Snapshot 2: Sign Up Page**

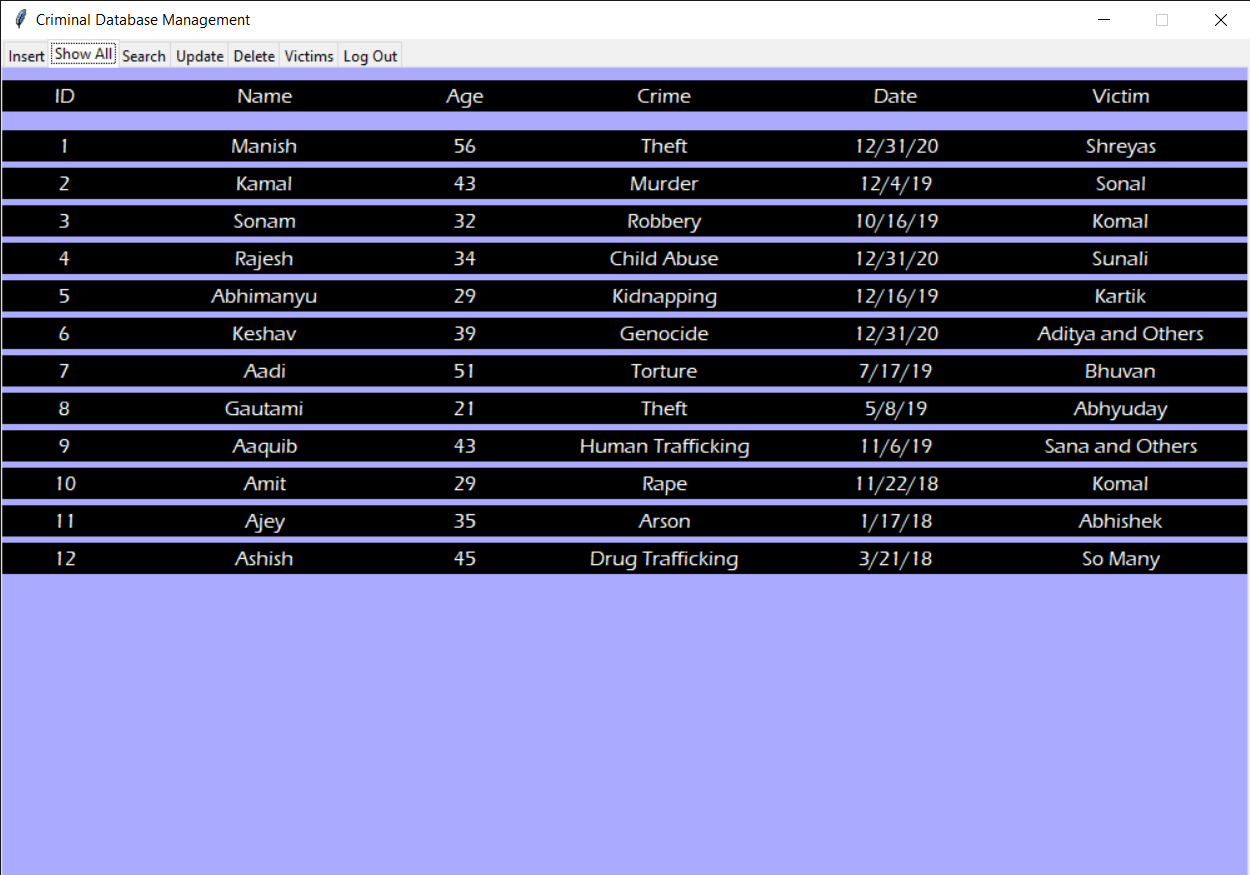
****

**Snapshot 3: Select Branch Page**

**Graphical user interface, application

Description automatically generated**

**Snapshot 4: Inserting a Record**

****

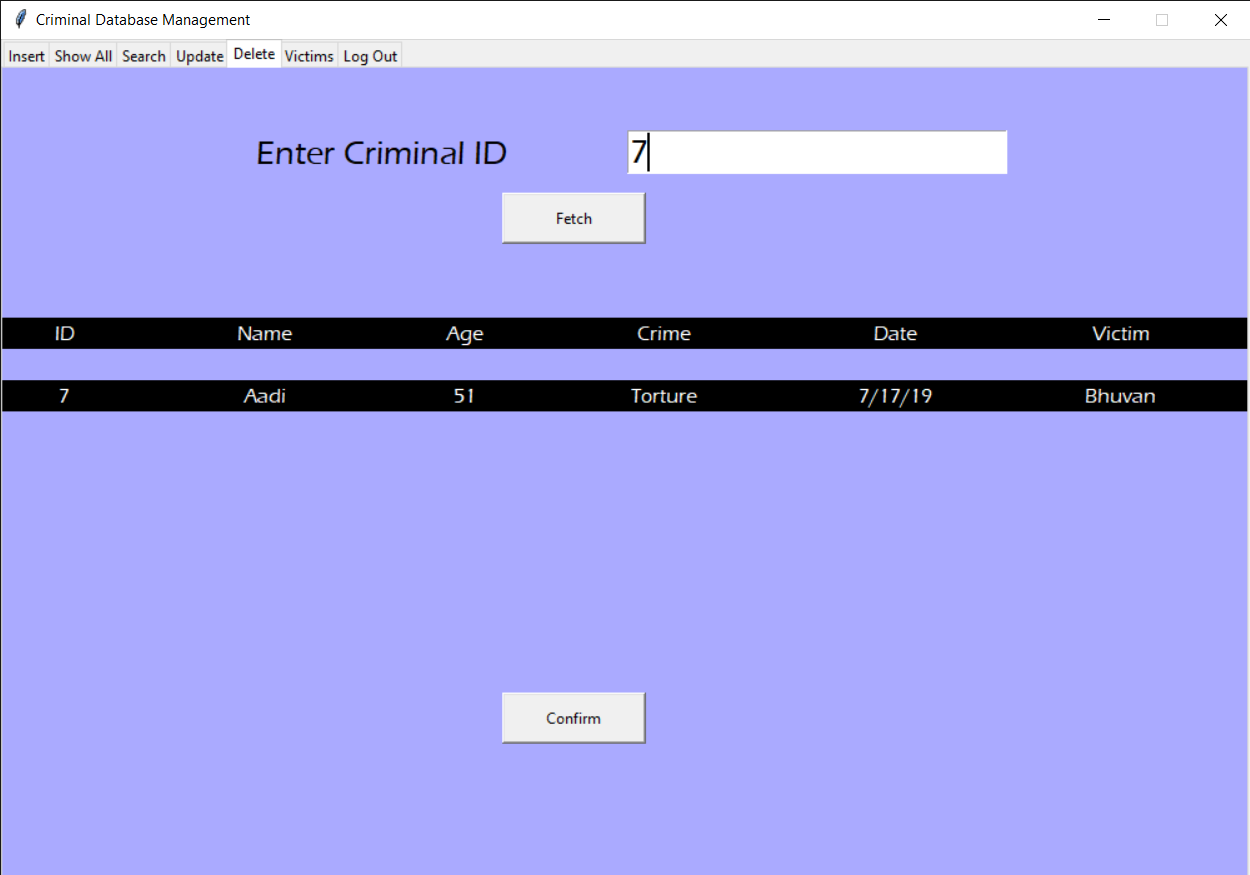
**Snapshot 5: Viewing all Records**

****

**Snapshot 6: Searching a Record**

****

**Snapshot 7: Updating a Record**

****

**Snapshot 8: Deleting a Record**

****

**Snapshot 9: Viewing Victims**

**CONCLUSION**

The development of this Criminal Database Management application is great improvement in accessing and holding the data comparing to manual work with so many paper. The computerization of the system speeds up the process.

The Criminal Database Management is fast, efficient and reliable, Avoids data redundancy and inconsistency. It contains all the functional features described in objective of the project.

**REFERENCES**

[1] Stack Overflow

[2] W3 Schools

[3] Codemy.com on YouTube

[4] Python 4 Everybody Book